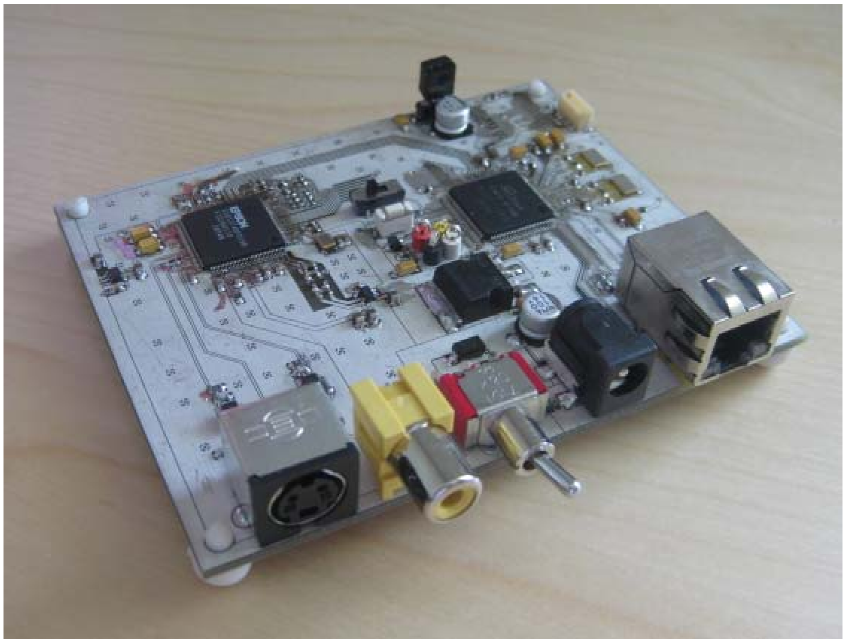
WIZnet 2010 iMCU Design Contest Entry #003183

UPnP Picture Viewer

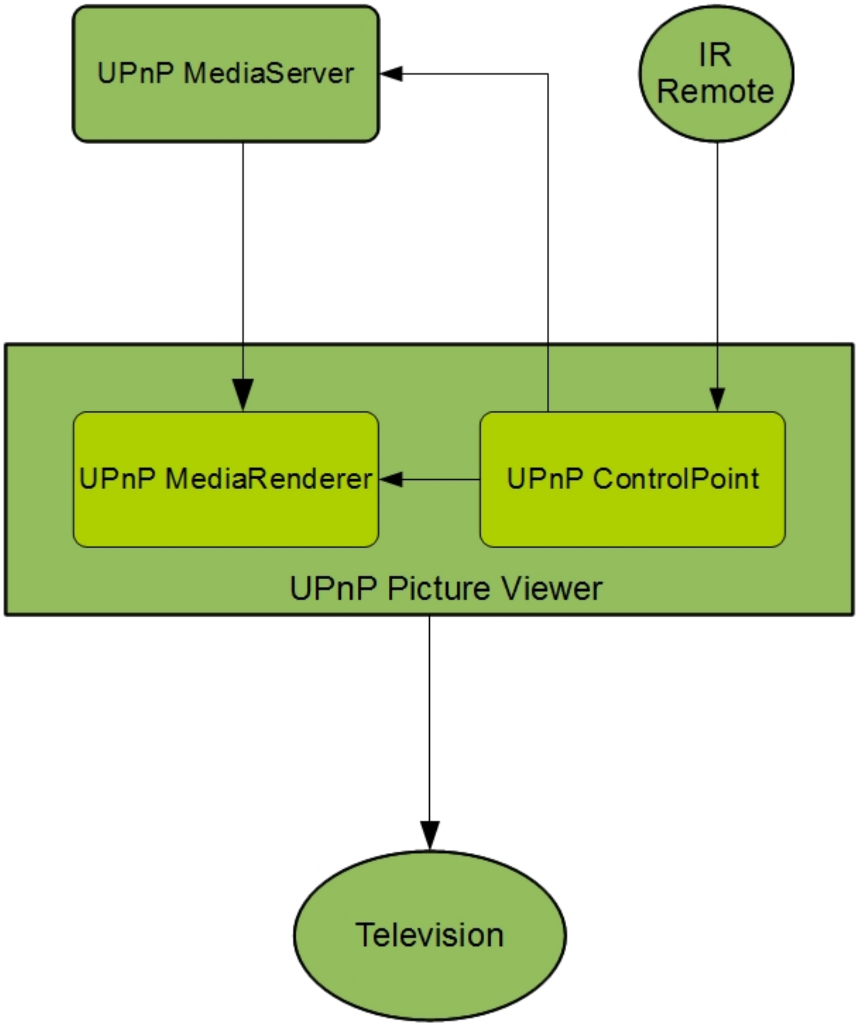
Abstract

When film photography was commonplace, sharing photos with friends and family usually involved sitting around flipping through a physical photo album. Today, film is all but obsolete and digital photography has taken over. With technological advances continually increasing storage capacity, people are taking more pictures than ever. But how does one share all of these photos with others? Printing hundreds to thousands of photos is expensive, and having a bunch of people huddled around a small computer screen is rather inconvenient. The UPnP Picture Viewer solves this problem by allowing the user to stream their digital photos across a network and display them on a standard television set.

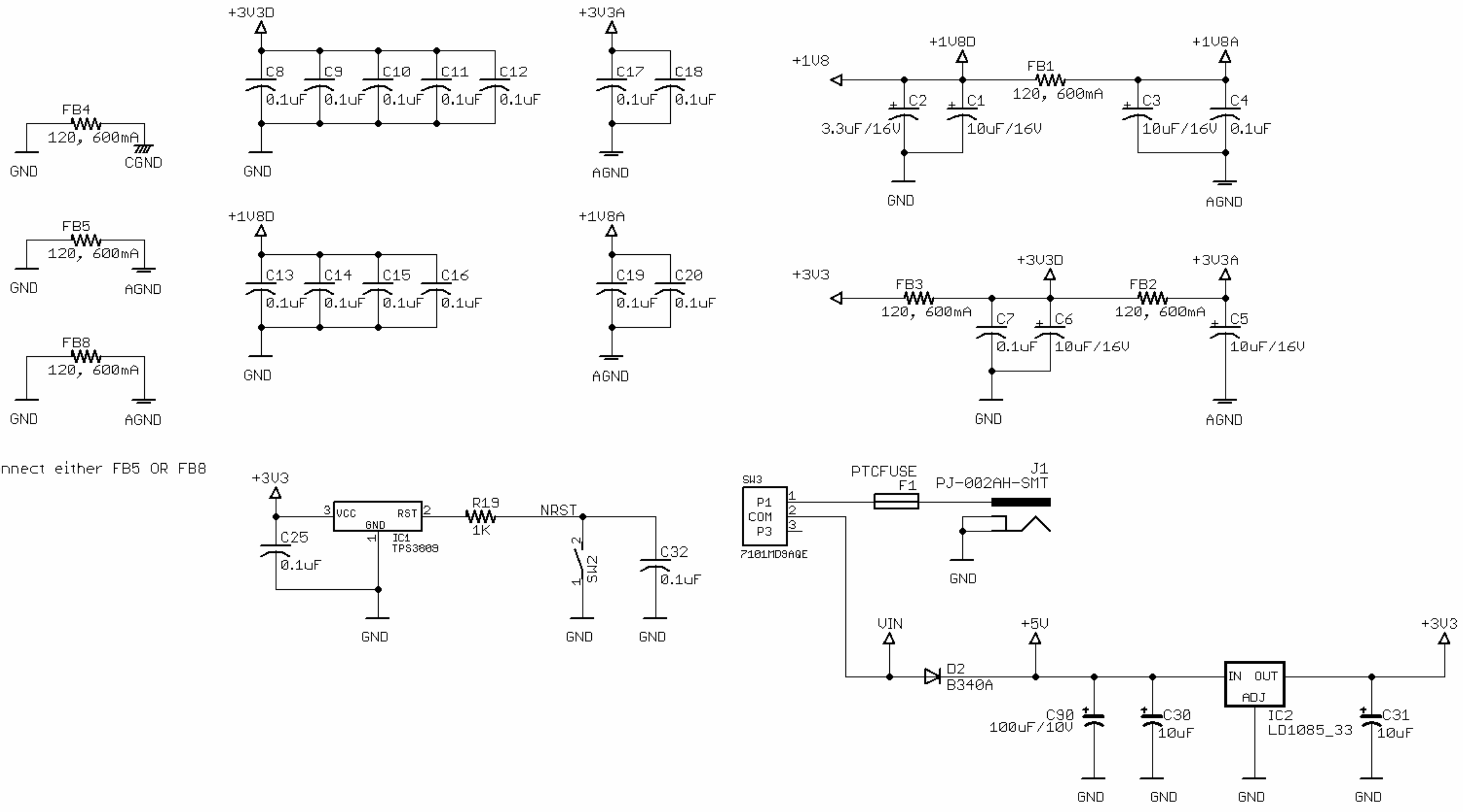
UPnP (Universal Plug and Play) is a set of networking protocols designed to simplify the sharing of data across a network. The UPnP Picture Viewer is a UPnP ControlPoint and MediaRenderer combo device. It interacts with one or more UPnP MediaServers that are located on a local network, finds any pictures the MediaServers are sharing, then allows a user to browse and view those pictures on any S-Video compatible display. Any modern Windows, Linux or Mac machine can run a UPnP MediaServer and many free software packages exist that implement that functionality.



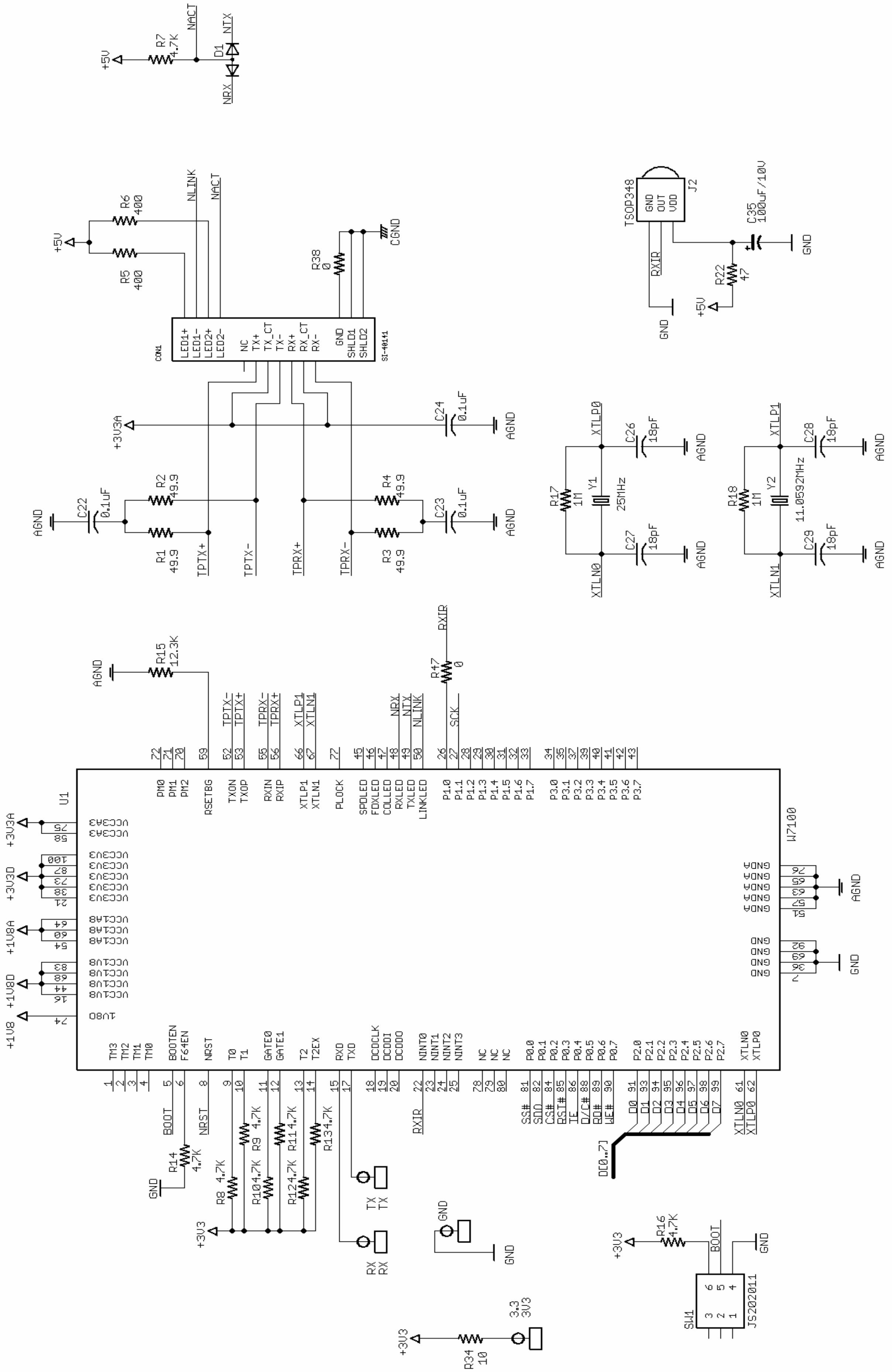
At its core, the UPnP Picture Viewer is powered by a WIZnet W7100 Internet MCU. The W7100 is an integrated networking solution, combining a hardware TCP/IP core and an 8051 compatible microcontroller. An Epson S1D13746 TV controller provides S-Video output capability and includes an integrated frame buffer. Control is possible via a universal IR remote control, which is demodulated by an onboard Vishay TSOP34838 IR receiver and decoded in software on the W7100.

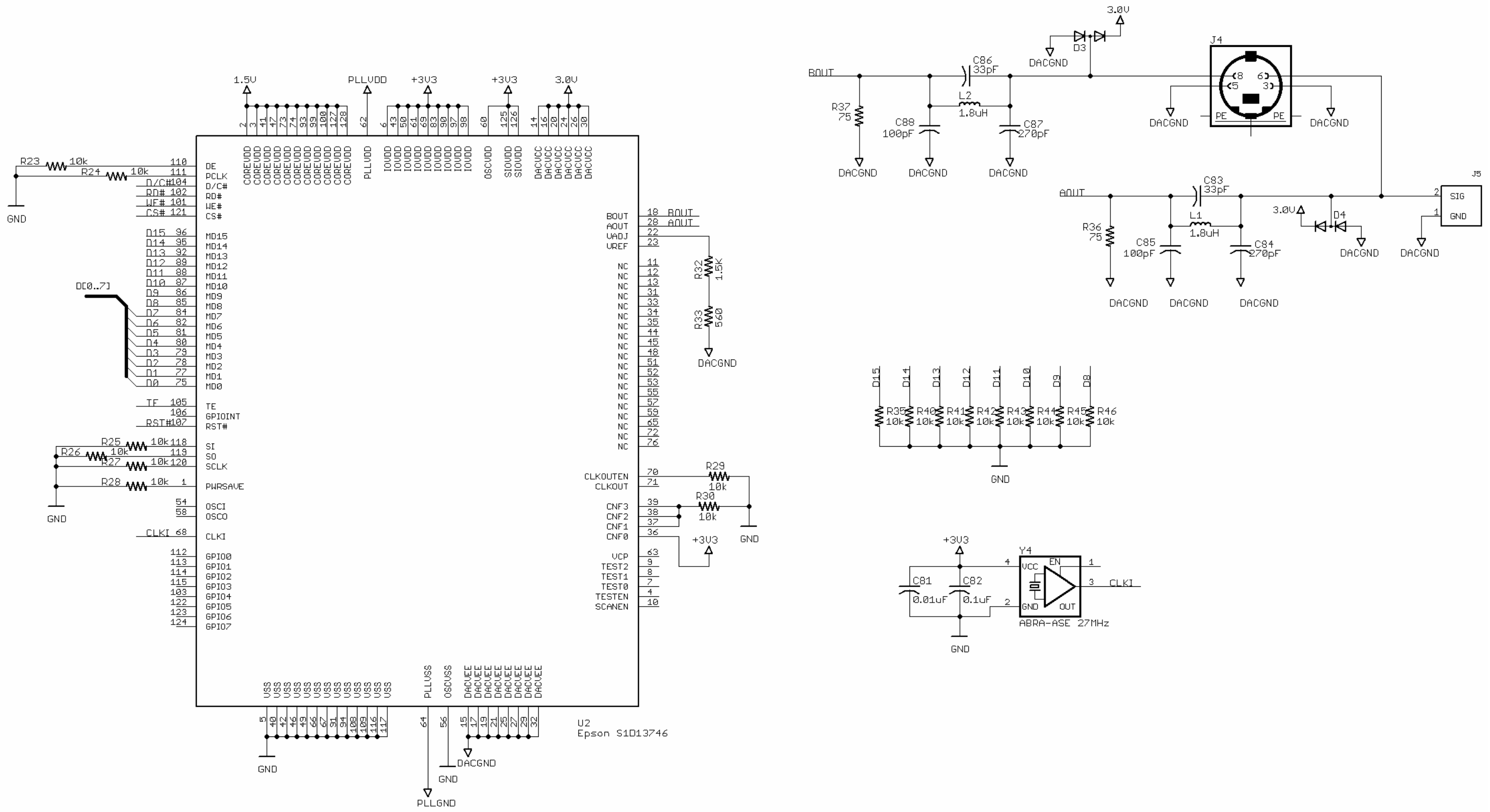


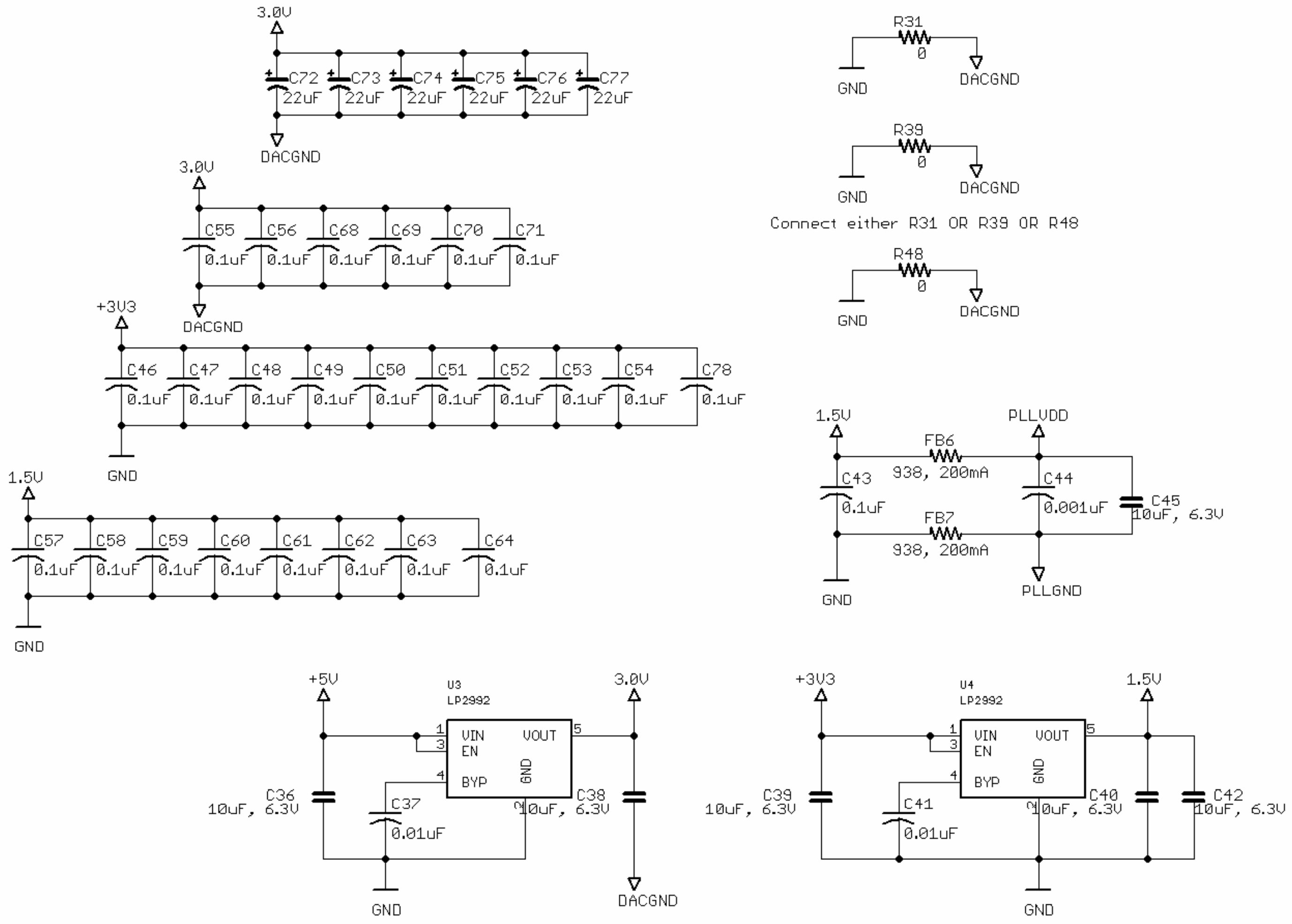
Although designed for manually viewing photos, the UPnP Picture Viewer can also be used as a digital picture frame when connected to a small LCD display and using the Viewer's automated slideshow feature. Small business owners may also find the device useful for displaying marketing or sales materials in a retail storefront.



Connect either FB5 OR FB8


U1
W7100
+5V
R7 4.7K
NACT
D1
NTX
NRX
+5V
R6 400
R5 400
NLINK
NACT
CON1
SI-40141
LED1+
LED1-
LED2+
LED2-
NC
TX+
TX_CT
TX-
RX+
RX_CT
RX-
GND
SHLD1
SHLD2
R38 0
CGND
TSOP348
GND
OUT
VDD
J2
C35 100uF/10V
RXIR
R22 47
+5V
GND
+3V3A
C24 0.1uF
AGND
C22 0.1uF
R2 49.9
R1 49.9
R4 49.9
R3 49.9
C23 0.1uF
TPTX+
TPTX-
TPRX+
TPRX-
XTLP0
C26 18pF
R17 1M
Y1 25MHz
C27 18pF
XTLN0
XTLP1
C28 18pF
R18 1M
Y2 11.0592MHz
C29 18pF
XTLN1
R15 12.3K
R47 0
RXIR
SCK
+3V3A
+3V3D
+1V8A
+1V8D
+1V8
VCC3A3
VCC3V3
VCC1A8
VCC1V8
1V8O
GNDA
GND
PM0
PM1
PM2
RSETBG
TXON
TXOP
RXIN
RXIP
XTLP1
XTLN1
PLOCK
SPDLED
FDXLED
COLLED
RXLED
TXLED
LINKLED
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P3.0
P3.1
P3.2
P3.3
P3.4
P3.5
P3.6
P3.7
TM3
TM2
TM1
TM0
BOOTEN
F64EN
NRST
T0
T1
GATE0
GATE1
T2
T2EX
RXD
TXD
DCDCLK
DCDDI
DCDDO
NINT0
NINT1
NINT2
NINT3
NC
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P2.6
P2.7
XTLN0
XTLP0
BOOT
NRST
RXIR
SS#
SDO
CS#
RST#
IE
D/C#
RD#
WE#
D0
D1
D2
D3
D4
D5
D6
D7
DE0..7]
XTLN0
XTLP0
R14 4.7K
GND
R8 4.7K
R9 4.7K
R10 4.7K
R11 4.7K
R12 4.7K
R13 4.7K
+3V3
TX
RX
GND
+3V3
R16 4.7K
BOOT
GND
SW1
JS202011
R34 10
3.3
3V3
+3V3


U2
Epson S1D13746
1.5V
PLLVDD
+3V3
+3V3
3.0V
2 3 41 47 73 74 93 99 100 127 128
COREVDD
62 PLLVDD
6 43 50 61 69 83 90 97 98
IOVDD
60 OSCVDD
125 126 SIOVDD
14 16 20 24 26 30
DACVCC
R23 10k
R24 10k
GND
110 DE
111 PCLK
D/C# 104 D/C#
RD# 102 RD#
WE# 101 WE#
CS# 121 CS#
D15 96 MD15
D14 95 MD14
D13 92 MD13
D12 89 MD12
D11 88 MD11
D10 87 MD10
D9 86 MD9
D8 85 MD8
D7 84 MD7
D6 82 MD6
D5 81 MD5
D4 80 MD4
D3 79 MD3
D2 78 MD2
D1 77 MD1
D0 75 MD0
D[0..7]
TE 105 TE
106 GPIOINT
RST# 107 RST#
R25 10k 118 SI
R26 10k 119 SO
R27 10k 120 SCLK
R28 10k 1 PWRSAVE
GND
54 OSCI
58 OSCO
CLKI 68 CLKI
112 GPIO0
113 GPIO1
114 GPIO2
115 GPIO3
103 GPIO4
122 GPIO5
123 GPIO6
124 GPIO7
VSS
5 40 42 46 49 66 67 91 94 108 109 116 117
GND
64 PLLVSS
PLLGND
56 OSCVSS
GND
DACVEE
15 17 19 21 25 27 29 32
DACGND
BOUT 18 BOUT
AOUT 28 AOUT
VADJ 22
VREF 23
R32 1.5K
R33 560
DACGND
NC 11 12 13 31 33 34 35 44 45 48 51 52 53 55 57 59 65 72 76
CLKOUTEN 70
CLKOUT 71
R29 10k
R30 10k
GND
CNF3 39
CNF2 38
CNF1 37
CNF0 36
+3V3
VCP 63
TEST2 9
TEST1 8
TEST0 7
TESTEN 4
SCANEN 10
BOUT
R37 75
DACGND
C88 100pF
DACGND
L2 1.8uH
C86 33pF
C87 270pF
DACGND
3.0V
D3
DACGND
J4
8 6 5 3
PE PE
DACGND
DACGND
AOUT
R36 75
DACGND
C85 100pF
DACGND
L1 1.8uH
C83 33pF
C84 270pF
DACGND
3.0V
D4
DACGND
J5
2 SIG
1 GND
DACGND
D15 D14 D13 D12 D11 D10 D9 D8
R35 10k
R40 10k
R41 10k
R42 10k
R43 10k
R44 10k
R45 10k
R46 10k
GND
+3V3
C81 0.01uF
C82 0.1uF
GND
Y4
4 VCC
EN 1
2 GND
OUT 3
CLKI
ABRA-ASE 27MHz



Connect either R31 OR R39 OR R48

```
int main()
{
     uint8 xdata txsize[MAX_SOCK_NUM] = {0,0,0,0,0,0,0,0};

     uint8 xdata rxsize[MAX_SOCK_NUM] = {0,0,0,0,0,0,0,0};
     uint8 xdata ret;
     uint32 xdata timeout=600000;

     printf("W7100 startup.\r\n");
     // initialize w7100 registers
     wiz_init_cpu();
     wiz_init_net();

     // initialize memory
     memset(pbuf, 0, sizeof(pbuf));
     memset(&upnp_object[0], 0, sizeof(upnp_object));
     memset(&upnp_service[0], 0, sizeof(upnp_service));
     memset(&upnp_state, 0, sizeof(upnp_state));

     // initialize display chip
     ret = epsonInit();
     if(ret)
          printf("S1D13746 initialized.\r\n");

     dispBootMsg(0);

     // get ip via dhcp
     while(!dhcpGetIP());

     dispBootMsg(1);
     dispBootMsg(2);

     // discover MediaServers
     while(!ssdpFindServices());

     // set socket memory to UPnP socket (TCP)
     txsize[UPNP_TCP_SOCKET] = 8;
     rxsize[UPNP_TCP_SOCKET] = 8;
     wiz_set_mem_size(txsize,rxsize);

     // retrieve service descriptions for content
     // directories
     upnpGetServiceDesc();

     dispBootMsg(3);

     // pause for a couple seconds after showing final boot
     // msg
     while(--timeout>0);

     // initalize interactive menu
     irInit();
     menuInit();
     menuUpdateServ();

     while(1)
     {
          // copy button input decoded from IR
          ir.latch = ir.input;

          // if in slideshow and flagged for next picture,
          //   move to next picture by simulating right
          //   button press
          if(menu.slideshow && menu.slideshow_next)
          {
               menu.slideshow_next = 0;
               ir.latch = IR_RIGHT;
          }

          // process button input
          menuRun();
     }

     return 1;
}
```